

can be associated with each attribute to optimize attribute retrieval from the database. As a default case, all attributes are retrieved when any one of an object's attributes are needed. However, the attribute retrieval list for any attribute can be edited to specify different attribute retrieval behavior. For example, a request for an Employee Id may cause the Photo attribute to be dropped from the attribute retrieval list on the Id attribute if that data resides in another table and is only infrequently used. Attribute retrieval lists are a performance feature that enable optimized data access by only doing JOINS and additional SELECT statements when the data returned by those actions is needed.

Performance is also enhanced by "just in time" data retrieval. By default, whenever an attribute value is read from the database, all of the other attributes for that instance are also read. However, Data Component Developers are permitted to modify the mapping information for a Data Component to define an attribute retrieval group for each attribute of a class that determines which other attribute values are returned when the requested attribute is read from the database. This makes it possible to avoid executing JOINS or SELECTs to retrieve data that may not be needed. For example, assume that a class, CPerson, has four attributes: Id, Name, Zip, and Photo, and the Photo attribute is mapped to a column in a different table from the others. The Data Component Developer may drop Photo from the group of attributes that are retrieved when either Id, Name, or Zip are read. A query is issued to get the Name and Id of an instance of CPerson where Id=10. Based on the attribute retrieval information, the run time engine retrieves only the values for the person.id, person.name, and person.zip attributes, thus avoiding an unnecessary join to return the photo attribute value as well.

If an object does not have an attribute in memory when an attempt is made to use that attribute, the object will issue a SELECT statement to retrieve the attribute from the database. "Just-in-time" attribute population allows the object to be populated with the minimal amount of information necessary for the application while still making any remaining information available when it is needed.

Lazy reads are also employed to enhance runtime performance. When a query is defined to identify objects for retrieval from the database, the SQL SELECT statement is not issued immediately. Queries are executed only after an attempt has been made to use or modify the resulting data.

Having described the embodiments consistent with the present invention, other embodiments and variations consistent with the present invention will be apparent to those skilled in the art. Therefore, the invention should not be viewed as limited to the disclosed embodiments but rather should be viewed as limited only by the spirit and scope of the appended claims.

What is claimed is:

1. A method for interfacing an object oriented software application with a relational database, comprising the steps of:

selecting an object model;  
generating a map of at least some relationships between schema in the database and the selected object model;  
employing the map to create at least one interface object associated with an object corresponding to a class associated with the object oriented software application; and

utilizing a runtime engine which invokes said at least one interface object with the object oriented application to access data from the relational database.

2. The method of claim 1 further including the step of mapping a class attribute to a table column.

3. The method of claim 1 further including the step of mapping a class attribute to a 1-1 relationship.

4. The method of claim 1 further including the step of mapping a class attribute to a 1-N relationship, where N is an integer that is greater than 1.

5. The method of claim 1 further including the step of mapping a class attribute to an N-N relationship, where N is an integer that is greater than 1.

6. The method of claim 1 further including the step of mapping class inheritance to rows within a table.

7. The method of claim 1 further including the step of mapping class inheritance across a plurality of tables.

8. The method of claim 1 further including the step of creating a plurality of said interface objects.

9. The method of claim 8 further including the step of creating at least one stateful interface object and at least one stateless interface object.

10. A computer program fixed on a computer-readable medium and adapted to operate on a computer to provide access to a relational database for an object oriented software application, comprising:

a mapping routine that generates a map of at least some relationships between schema in the database and a selected object model;

a code generator that employs said map to create at least one interface object associated with an object corresponding to a class associated with the object oriented software application; and

a runtime engine that invokes said at least one interface object to access data from the relational database.

11. The program of claim 10 wherein said mapping routine is operative to map a class attribute to a table column.

12. The program of claim 10 wherein said mapping routine is operative to map a class attribute to a 1-1 relationship.

13. The program of claim 10 wherein said mapping routine is operative to map a class attribute to a 1-N relationship, where N is an integer that is greater than 1.

14. The program of claim 10 wherein said mapping routine is operative to map a class attribute to an N-N relationship, where N is an integer that is greater than 1.

15. The program of claim 10 wherein said mapping routine is operative to map class inheritance to rows within a table.

16. The program of claim 10 wherein said mapping routine is operative to map class inheritance across a plurality of tables.

17. The program of claim 10 wherein said code generator is operative to create a plurality of said interface objects.

18. The program of claim 17 wherein said code generator is operative to create at least one stateful interface object and at least one stateless interface object.

\* \* \* \* \*